IN THE UNITED STATES DISTRICT COURT
FOR THE WESTERN DISTRICT OF TEXAS
AUSTIN DIVISION

LUCIO DEVELOPMENT LLC,	§	
	§	
Plaintiff,	§	Case No: 1:17-cv-1149
	§	
vs.	§	PATENT CASE
	§	
CIRRUS LOGIC, INC.,	§	
	§	
Defendant.	§	
	§	

COMPLAINT

Plaintiff Lucio Development LLC ("Plaintiff" or "Lucio") files this Complaint against Cirrus Logic, Inc. ("Defendant" or "Cirrus") for infringement of United States Patent No. 7,069,546 (hereinafter "the '546 Patent").

PARTIES AND JURISDICTION

1. This is an action for patent infringement under Title 35 of the United States Code. Plaintiff is seeking injunctive relief as well as damages.

2. Jurisdiction is proper in this Court pursuant to 28 U.S.C. §§ 1331 (Federal Question) and 1338(a) (Patents) because this is a civil action for patent infringement arising under the United States patent statutes.

3. Plaintiff is a Texas limited liability company with its office address at 555 Republic Dr., Suite 200, Plano, Texas 75074.

4. On information and belief, Defendant is a Delaware corporation with a place of business at 800 W. 6th Street, Austin, Texas 78701 and a registered agent for service of process at CT Corporation System, 1999 Bryan St, Suite 900, Dallas, TX 75201.

5. This Court has personal jurisdiction over Defendant because Defendant has committed, and continues to commit, acts of infringement in this District, has conducted business in this District, and/or has engaged in continuous and systematic activities in this District.

6. On information and belief, Defendant's instrumentalities that are alleged herein to infringe were and continue to be used, imported, offered for sale, and/or sold in this District.

VENUE

7. Venue is proper in this District pursuant to 28 U.S.C. §1400(b) because acts of infringement are occurring in this District and Defendant has a regular and established place of business in this District. For instance, on information and belief, Defendant has a regular and established place of business at 800 W. 6th Street, Austin, Texas 78701.

COUNT I
(INFRINGEMENT OF UNITED STATES PATENT NO. 7,069,546)

8. Plaintiff incorporates paragraphs 1 through 7 herein by reference.

9. This cause of action arises under the patent laws of the United States and, in particular, under 35 U.S.C. §§ 271, *et seq*.

10. Plaintiff is the owner by assignment of the '546 Patent with sole rights to enforce the '546 Patent and sue infringers.

11. A copy of the '546 Patent, titled "Generic Framework for Embedded Software Development," is attached hereto as Exhibit A.

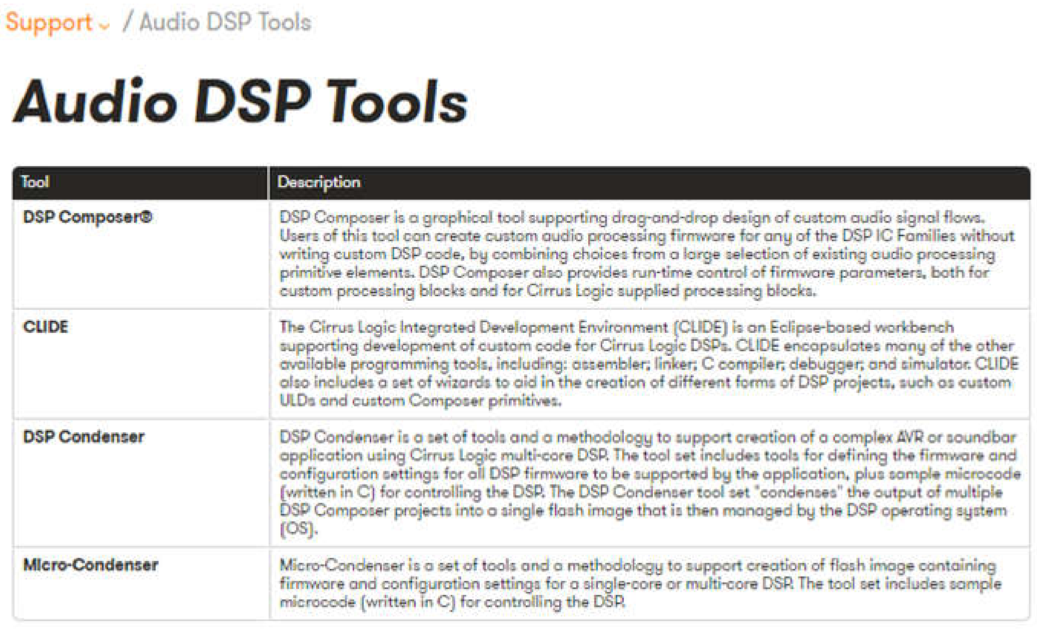
12. The '546 Patent is valid, enforceable, and was duly issued in full compliance with Title 35 of the United States Code.

13. On information and belief, Defendant has infringed and continues to infringe one or more claims, including at least Claim 1, of the '546 Patent by making, using, importing,

selling, and/or offering for sale a software platform for embedded software development, which is covered by at least Claim 1 of the ‘546 Patent. Defendant has infringed and continues to infringe the ‘546 Patent directly in violation of 35 U.S.C. § 271.

14. Defendant, sells, offers to sell, and/or uses embedded software development packages including, without limitation, Cirrus Audio DSP Tools, and any similar products (“Product”), which infringe at least Claim 1 of the ‘546 Patent.

15. The Product is a framework (e.g., a software development kit) that is configured to create embedded software for multiple hardware modules. For example, the Product is a programmable software development kit (SDK) for multiple hardware such as CS4953xx, CS4970x4, CS485xx, CS470xx, and the CS498xx multicore DSPs. Defendant and/or its customers specifically use Cirrus Audio DSP Tools to develop code, compile and debug on the target devices. Certain elements of this limitation are illustrated in the screenshots below and in the screenshots referenced in connection with other elements herein.


Support ⌄ / Audio DSP Tools

Audio DSP Tools

Tool	Description
DSP Composer®	DSP Composer is a graphical tool supporting drag-and-drop design of custom audio signal flows. Users of this tool can create custom audio processing firmware for any of the DSP IC Families without writing custom DSP code, by combining choices from a large selection of existing audio processing primitive elements. DSP Composer also provides run-time control of firmware parameters, both for custom processing blocks and for Cirrus Logic supplied processing blocks.
CLIDE	The Cirrus Logic Integrated Development Environment (CLIDE) is an Eclipse-based workbench supporting development of custom code for Cirrus Logic DSPs. CLIDE encapsulates many of the other available programming tools, including: assembler; linker; C compiler; debugger; and simulator. CLIDE also includes a set of wizards to aid in the creation of different forms of DSP projects, such as custom ULDs and custom Composer primitives.
DSP Condenser	DSP Condenser is a set of tools and a methodology to support creation of a complex AVR or soundbar application using Cirrus Logic multi-core DSP. The tool set includes tools for defining the firmware and configuration settings for all DSP firmware to be supported by the application, plus sample microcode (written in C) for controlling the DSP. The DSP Condenser tool set "condenses" the output of multiple DSP Composer projects into a single flash image that is then managed by the DSP operating system (OS).
Micro-Condenser	Micro-Condenser is a set of tools and a methodology to support creation of flash image containing firmware and configuration settings for a single-core or multi-core DSP. The tool set includes sample microcode (written in C) for controlling the DSP.

Source: https://www.cirrus.com/support/dsp/

8.1 Overview

Cirrus Logic provides the customer with the DSP Condenser application to implement the design capabilities described in Section P.1. The DSP Condenser application has the following features:

- Customer specifies the DSP features such as the decoding modules to be used in the customer application.
- Application matches the appropriate Cirrus firmware with the customer-selected processing and decoding modules and loads it on the customer's Flash device for testing on Cirrus Logic development boards or on the customer's system.

How DSP Condenser provides the control code to the DSP is described in Figure 8-1.

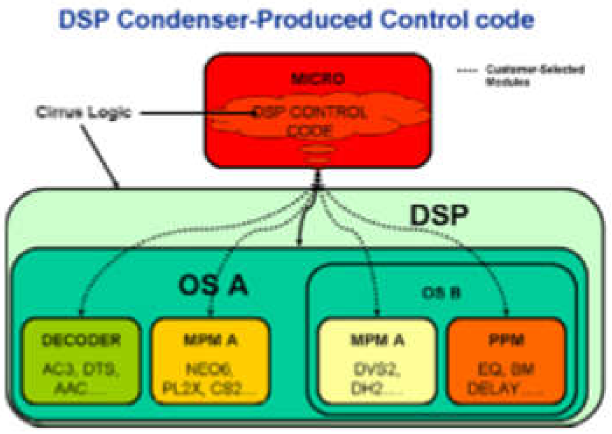


Figure 8-1. DSP Condenser-produced Control Code

Source: https://www.cirrus.com/support/dsp/

DSP Condenser	DSP Condenser is a set of tools and a methodology to support creation of a complex AVR or soundbar application using Cirrus Logic multi-core DSP. The tool set includes tools for defining the firmware and configuration settings for all DSP firmware to be supported by the application, plus sample microcode (written in C) for controlling the DSP. The DSP Condenser tool set "condenses" the output of multiple DSP Composer projects into a single flash image that is then managed by the DSP operating system (OS).

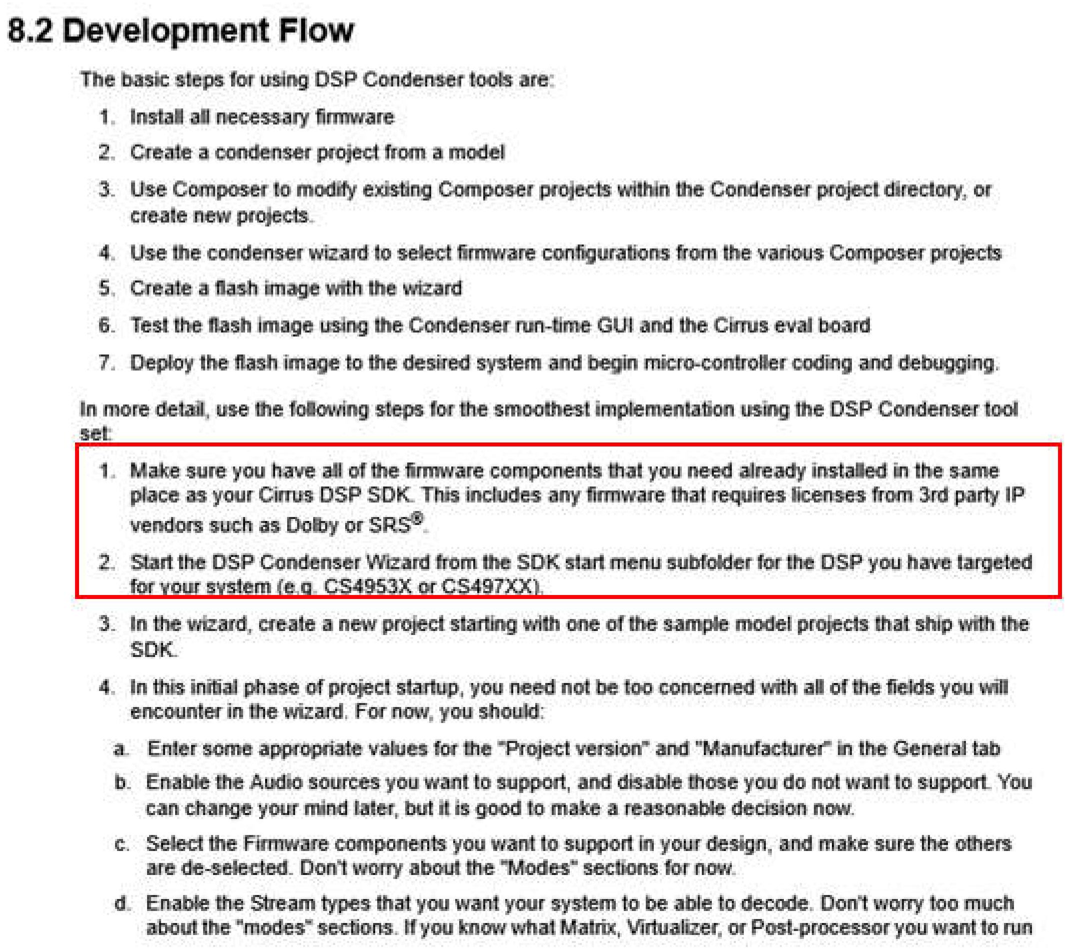
8.1.1 Purpose of DSP Condenser

DSP Condenser is a set of tools and a methodology designed to make it easier to develop a Cirrus DSP-based audio system. Using Condenser tools, the system designer can create a flash image that is directly accessible by the DSP operating system, and which contains all of the DSP firmware and most of the firmware configurations necessary to implement the desired DSP audio processing flow, with minimal control required by the system micro-controller.

Conceptually, DSP condenser allows the system designer to:

- Choose, at a high level, what firmware components to include in the design, including decoder and other audio processing algorithms
- Use the DSP Composer graphical tool to tune the parameters to any or all of the firmware components in the design
- "Condense" the chosen firmware components and configurations into a single flash image, ready to be programmed and embedded in the system with the DSP
- Test the flash image using the same high-level API that the system micro-controller should use in the final system

Source: https://www.cirrus.com/support/dsp/



8.2 Development Flow

The basic steps for using DSP Condenser tools are:

1. Install all necessary firmware
2. Create a condenser project from a model
3. Use Composer to modify existing Composer projects within the Condenser project directory, or create new projects.
4. Use the condenser wizard to select firmware configurations from the various Composer projects
5. Create a flash image with the wizard
6. Test the flash image using the Condenser run-time GUI and the Cirrus eval board
7. Deploy the flash image to the desired system and begin micro-controller coding and debugging.

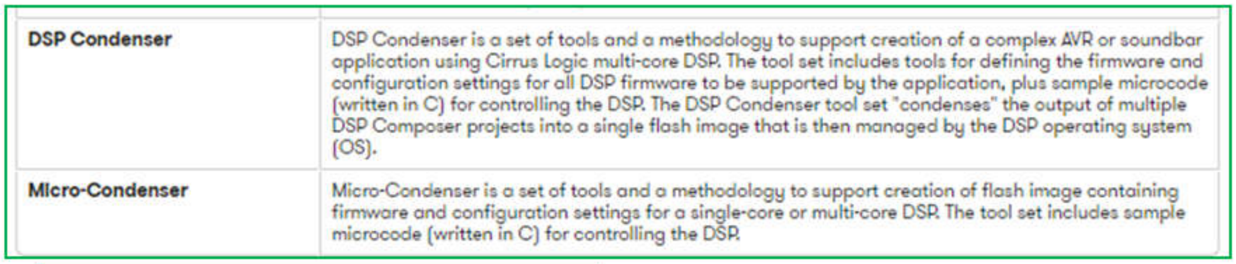
In more detail, use the following steps for the smoothest implementation using the DSP Condenser tool set:

1. Make sure you have all of the firmware components that you need already installed in the same place as your Cirrus DSP SDK. This includes any firmware that requires licenses from 3rd party IP vendors such as Dolby or SRS®.
2. Start the DSP Condenser Wizard from the SDK start menu subfolder for the DSP you have targeted for your system (e.g. CS4953X or CS497XX).
3. In the wizard, create a new project starting with one of the sample model projects that ship with the SDK.
4. In this initial phase of project startup, you need not be too concerned with all of the fields you will encounter in the wizard. For now, you should:
   a. Enter some appropriate values for the "Project version" and "Manufacturer" in the General tab
   b. Enable the Audio sources you want to support, and disable those you do not want to support. You can change your mind later, but it is good to make a reasonable decision now.
   c. Select the Firmware components you want to support in your design, and make sure the others are de-selected. Don't worry about the "Modes" sections for now.
   d. Enable the Stream types that you want your system to be able to decode. Don't worry too much about the "modes" sections. If you know what Matrix, Virtualizer, or Post-processor you want to run

Source: https://www.cirrus.com/support/dsp/

16. The Product provides one or more generic application handler programs. For example, Cirrus Audio DSP Tools provides one or more generic application handler programs such as control code, functions and data structures common and uniform across all supported hardware (such as CS4953xx, CS4970x4, CS485xx, CS470xx, and the CS498xx multicore DSPs). Such programs are generated using, for example, DSP Condenser and Micro-Condenser. The generic programs comprise computer program code for performing generic application functions common to multiple types of hardware modules used in a communication environment (e.g., the generic code provides common and generic functions to multiple hardware modules, as previously identified in paragraph 15. Certain elements of this limitation

are illustrated in the screenshots below and in the screenshots referenced in connection with other elements herein.



DSP Condenser	DSP Condenser is a set of tools and a methodology to support creation of a complex AVR or soundbar application using Cirrus Logic multi-core DSP. The tool set includes tools for defining the firmware and configuration settings for all DSP firmware to be supported by the application, plus sample microcode (written in C) for controlling the DSP. The DSP Condenser tool set "condenses" the output of multiple DSP Composer projects into a single flash image that is then managed by the DSP operating system (OS).
Micro-Condenser	Micro-Condenser is a set of tools and a methodology to support creation of flash image containing firmware and configuration settings for a single-core or multi-core DSP. The tool set includes sample microcode (written in C) for controlling the DSP.

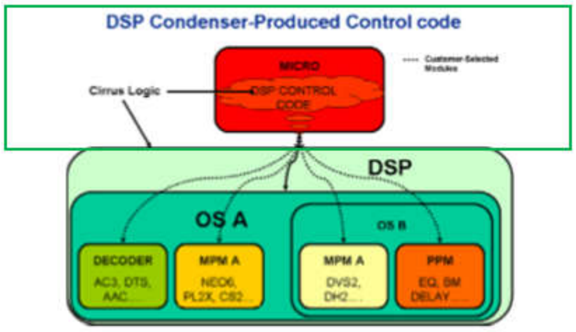
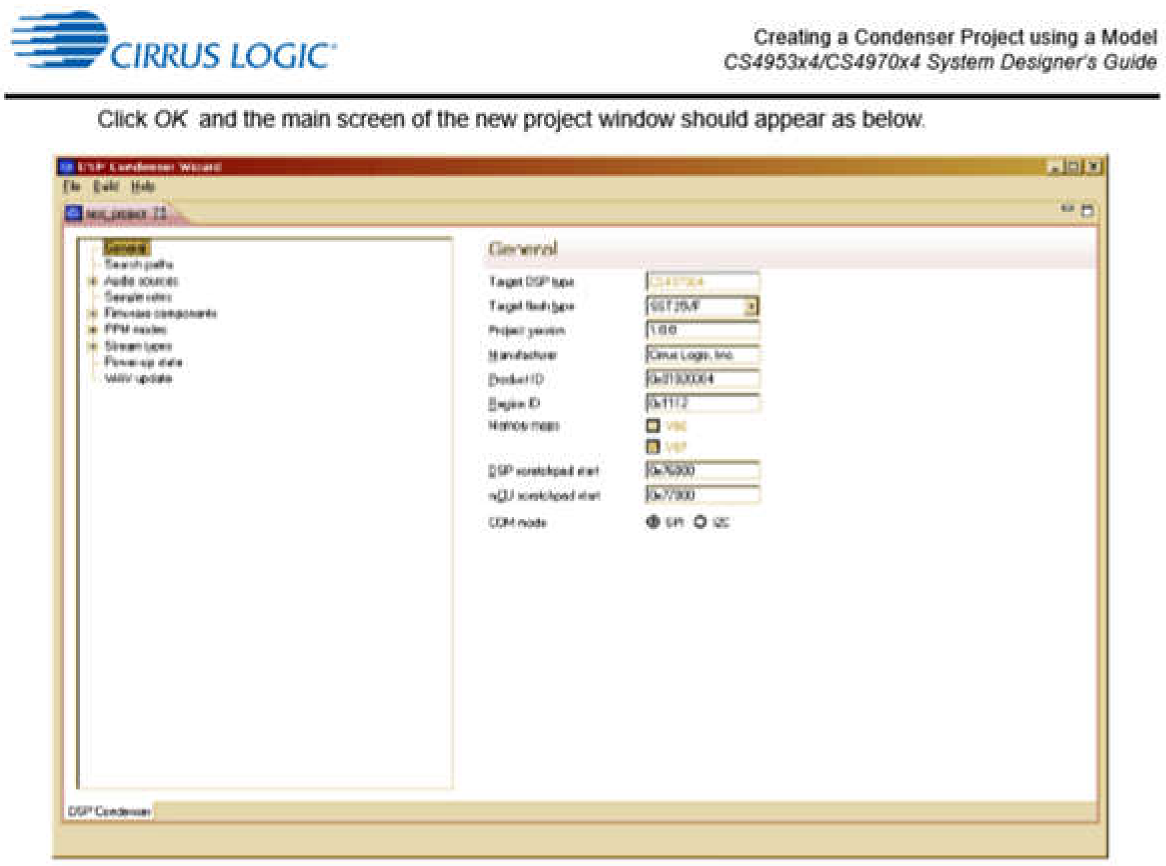


Figure 8-1. DSP Condenser-produced Control Code

<u>Source</u>: https://www.cirrus.com/support/dsp/


Creating a Condenser Project using a Model
CS4953x4/CS4970x4 System Designer's Guide

Click *OK* and the main screen of the new project window should appear as below.

General

Target DSP type	CS4970x4
Target flash type	SST 25VF
Project version	1.0.0
Manufacturer	Cirrus Logic, Inc.
Product ID	0x01300004
Region ID	0x1112
Memory maps	VBC VBP
DSP scratchpad start	0x76000
uCU scratchpad start	0x77000
COM mode	[✓ SPI] I2C

Source: https://www.cirrus.com/support/dsp/

17. The Product includes generating specific application handler code to associate the generic functions with the specific functions at a device driver for at least one of the types of hardware modules. For example, in addition to the generic application handler code, Cirrus Audio DSP Tools also includes specific application handler code that is specific to the application (such as OS A and OS B firmware modules for decoder, MPM and PPM) and specific to particular hardware (such as particular Cirrus Logic supported boards). Certain elements of this limitation are illustrated in the screenshots below and in the screenshots referenced in connection with other elements herein.

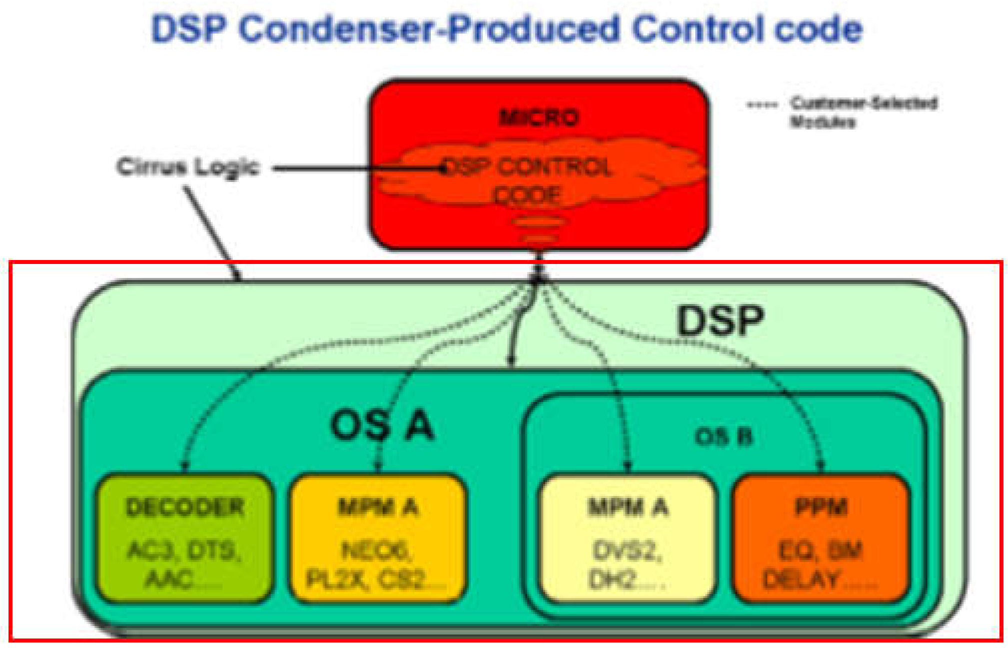


Figure 8-1. DSP Condenser-produced Control Code

Source: https://www.cirrus.com/support/dsp/

7.2 CS4953x4/CS4970x4 Firmware

7.2.1 Firmware Modules

The CS4953x4/CS4970x4 DSP provides both standard and advanced audio decoding and processing using factory-supplied firmware modules that are downloaded to the device and executed on one of the DSP cores. Each module provides a specific processing functionality. For instance, the DTS module provides DTS® decoding, the Dolby Headphone® module provides Dolby Headphone processing capability and the Bass Management module provides Bass Management processing.

7.2.2 Overlay Architecture

For memory management reasons, modules are grouped into overlays, which are separately loadable. Most overlays have a one-to-one correspondence with modules. Post processing overlays (PPMs) generally have a one-to-many correspondence: one overlay contains many modules.

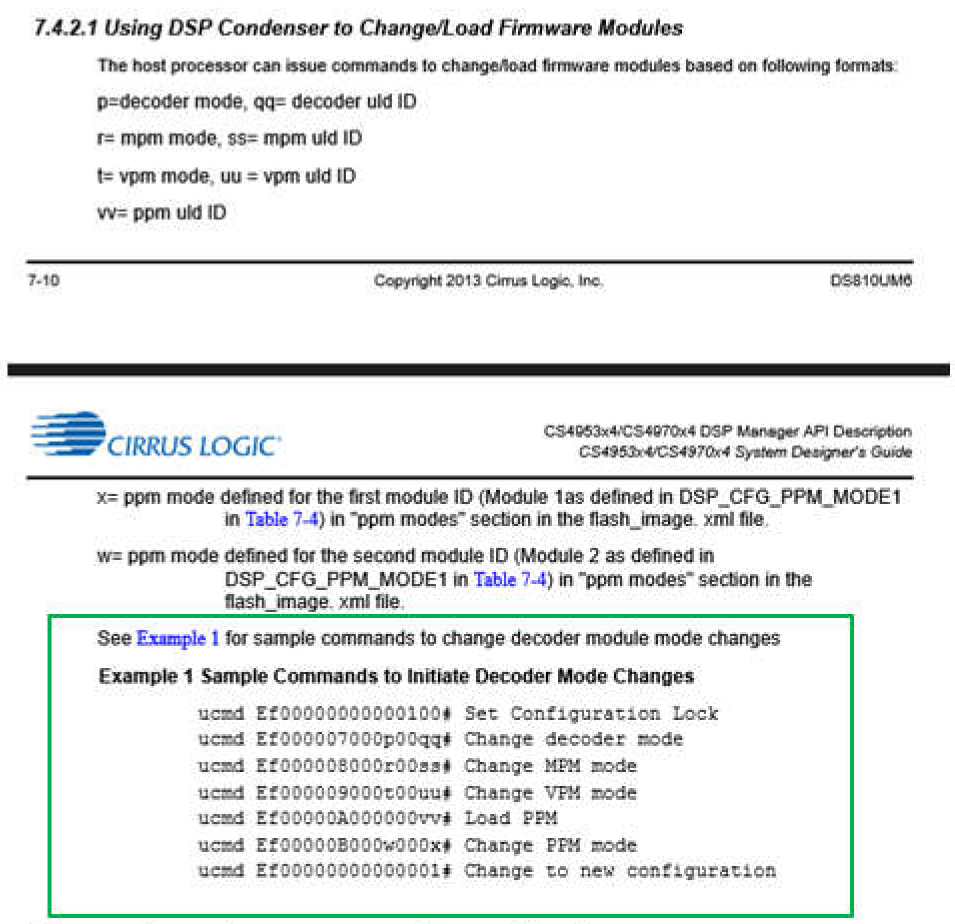
Overlays are divided into classes, based on the type of audio processing the modules in that overlay class do. In the CS49700, the following overlay classes are defined:

- OS - operating system functions
- DEC - decoder modules (for example, AC3, DTDHDHRA)
- MPM - matrix processing modules (for example, PL2, NEO6, COMS2)
- VPM - virtualizer modules (for example, Dolby Headphone)
- PPM - post processor modules

At any given moment, only one overlay of each class can be loaded in the DSP (the OS actually is loaded into both DSP cores, DSP A and DSP B).

Source: https://www.Cirrus Logic.com/dgdl/Cirrus Logic-Cirrus Audio DSP Tools-Digital-Application-Virtual-Engineer-for-XMC-MCUs-BC-v01_00-EN.pdf?fileId=5546d4624e765da5014edee86bb71ac8

18. The Product generates specific application handler code and defines a specific element in the specific code to be handled by one of the generic application functions for that hardware module. For example**,** Cirrus Audio DSP Tools generates system-specific application handler code by defining a specific element such as functions and data structures corresponding to specific hardware components (such as MPM, VPM, PPM modes) that extend or otherwise connect the system-specific application handler code to the functions and data structures defined and made available by the condenser. When specific functions are written for handling defined specific elements, the specific functions must be registered. Cirrus Logic C-Compiler, for example, contains the instructions necessary for registering the required functions. Certain elements of this limitation are illustrated in the screenshots below and in the screenshots referenced in connection with other elements herein.



7.4.2.1 Using DSP Condenser to Change/Load Firmware Modules

The host processor can issue commands to change/load firmware modules based on following formats:

p=decoder mode, qq= decoder uld ID

r= mpm mode, ss= mpm uld ID

t= vpm mode, uu = vpm uld ID

vv= ppm uld ID

7-10 Copyright 2013 Cirrus Logic, Inc. DS810UM6

CIRRUS LOGIC

CS4953x4/CS4970x4 DSP Manager API Description
CS4953x4/CS4970x4 System Designer's Guide

x= ppm mode defined for the first module ID (Module 1as defined in DSP_CFG_PPM_MODE1 in Table 7-4) in "ppm modes" section in the flash_image. xml file.

w= ppm mode defined for the second module ID (Module 2 as defined in DSP_CFG_PPM_MODE1 in Table 7-4) in "ppm modes" section in the flash_image. xml file.

See Example 1 for sample commands to change decoder module mode changes

Example 1 Sample Commands to Initiate Decoder Mode Changes

```
ucmd Ef00000000000100# Set Configuration Lock
ucmd Ef000007000p00qq# Change decoder mode
ucmd Ef000008000r00ss# Change MPM mode
ucmd Ef000009000t00uu# Change VPM mode
ucmd Ef00000A000000vv# Load PPM
ucmd Ef00000B000w000x# Change PPM mode
ucmd Ef00000000000001# Change to new configuration
```

<u>Source</u>: https://www.cirrus.com/support/dsp/

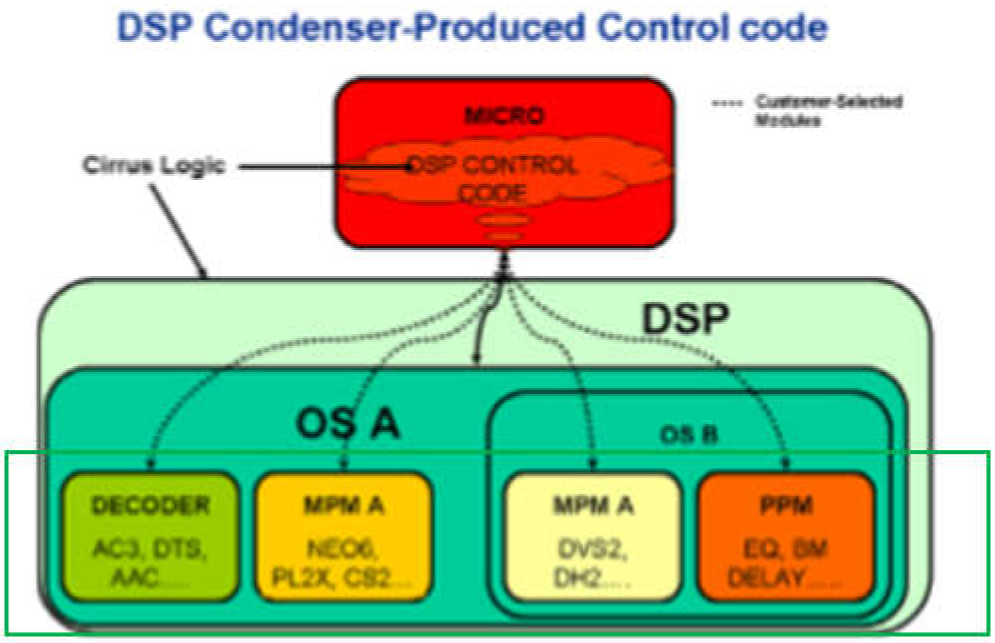
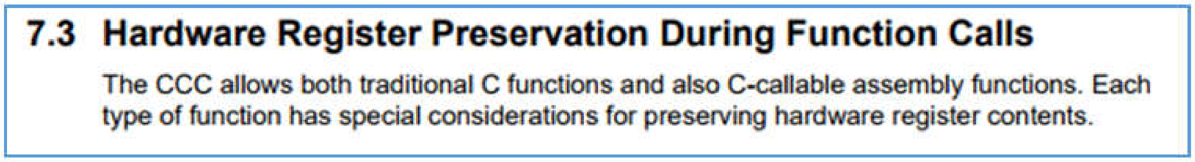


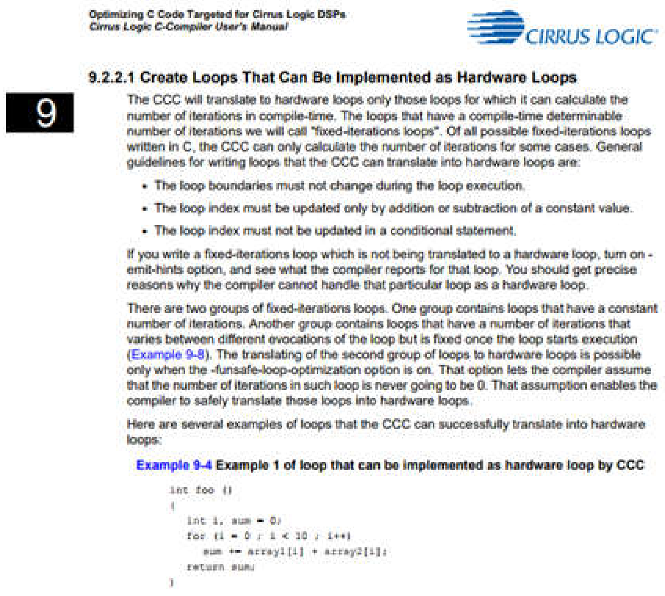
Figure 8-1. DSP Condenser-produced Control Code

<u>Source</u>: https://www.cirrus.com/support/dsp/



7.3 Hardware Register Preservation During Function Calls

The CCC allows both traditional C functions and also C-callable assembly functions. Each type of function has special considerations for preserving hardware register contents.



Optimizing C Code Targeted for Cirrus Logic DSPs
Cirrus Logic C-Compiler User's Manual

9

9.2.2.1 Create Loops That Can Be Implemented as Hardware Loops

The CCC will translate to hardware loops only those loops for which it can calculate the number of iterations in compile-time. The loops that have a compile-time determinable number of iterations we will call "fixed-iterations loops". Of all possible fixed-iterations loops written in C, the CCC can only calculate the number of iterations for some cases. General guidelines for writing loops that the CCC can translate into hardware loops are:

- The loop boundaries must not change during the loop execution.
- The loop index must be updated only by addition or subtraction of a constant value.
- The loop index must not be updated in a conditional statement.

If you write a fixed-iterations loop which is not being translated to a hardware loop, turn on -emit-hints option, and see what the compiler reports for that loop. You should get precise reasons why the compiler cannot handle that particular loop as a hardware loop.

There are two groups of fixed-iterations loops. One group contains loops that have a constant number of iterations. Another group contains loops that have a number of iterations that varies between different evocations of the loop but is fixed once the loop starts execution (Example 9-8). The translating of the second group of loops to hardware loops is possible only when the -funsafe-loop-optimization option is on. That option lets the compiler assume that the number of iterations in such loop is never going to be 0. That assumption enables the compiler to safely translate those loops into hardware loops.

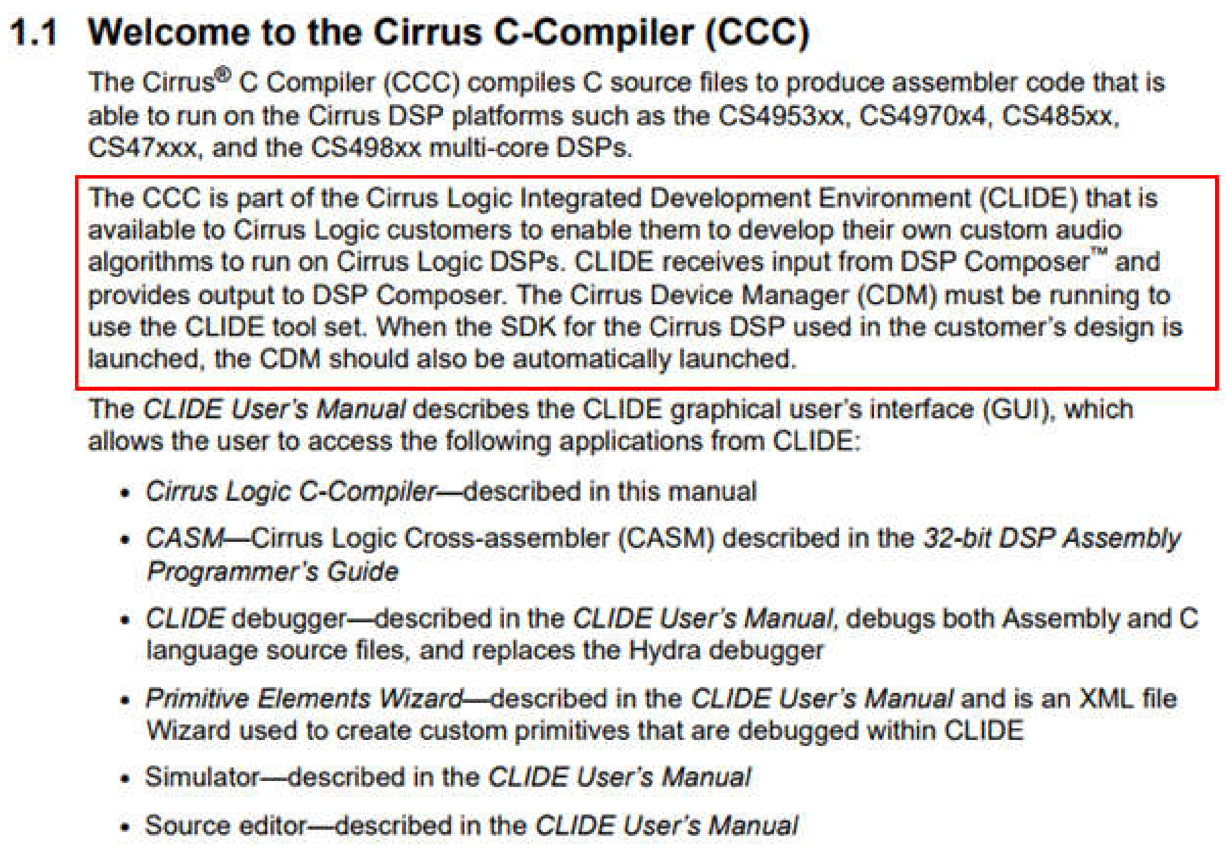
Here are several examples of loops that the CCC can successfully translate into hardware loops:

Example 9-4 Example 1 of loop that can be implemented as hardware loop by CCC

```
int foo ()
{
  int i, sum = 0;
  for (i = 0 ; i < 10 ; i++)
    sum += array1[i] + array2[i];
  return sum;
}
```

Source: https://d3uzseaevmutz1.cloudfront.net/pubs/manual/cirrus_c_compiler_um15.pdf

19. When a specific application is needed for a particular hardware, the generic functions and the specific functions are compiled together to yield a machine readable code. Cirrus Logic and/or its customers compile the generic functions and the specific functions using Cirrus Logic Integrated Development Environment (CLIDE) and/or any other compiling IDE supported by Cirrus Logic. Certain elements of this limitation are illustrated in the screenshots below and in the screenshots referenced in connection with other elements herein.



1.1 Welcome to the Cirrus C-Compiler (CCC)

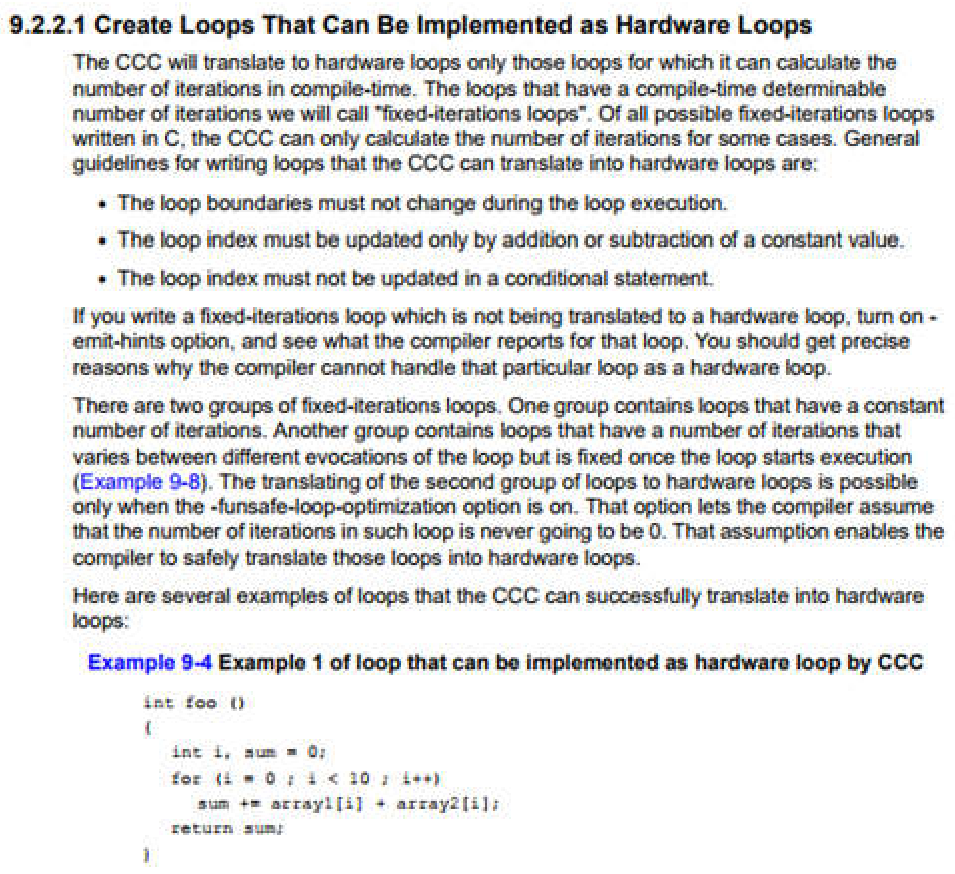
The Cirrus® C Compiler (CCC) compiles C source files to produce assembler code that is able to run on the Cirrus DSP platforms such as the CS4953xx, CS4970x4, CS485xx, CS47xxx, and the CS498xx multi-core DSPs.

The CCC is part of the Cirrus Logic Integrated Development Environment (CLIDE) that is available to Cirrus Logic customers to enable them to develop their own custom audio algorithms to run on Cirrus Logic DSPs. CLIDE receives input from DSP Composer™ and provides output to DSP Composer. The Cirrus Device Manager (CDM) must be running to use the CLIDE tool set. When the SDK for the Cirrus DSP used in the customer's design is launched, the CDM should also be automatically launched.

The *CLIDE User's Manual* describes the CLIDE graphical user's interface (GUI), which allows the user to access the following applications from CLIDE:

- *Cirrus Logic C-Compiler*—described in this manual
- *CASM*—Cirrus Logic Cross-assembler (CASM) described in the *32-bit DSP Assembly Programmer's Guide*
- *CLIDE* debugger—described in the *CLIDE User's Manual*, debugs both Assembly and C language source files, and replaces the Hydra debugger
- *Primitive Elements Wizard*—described in the *CLIDE User's Manual* and is an XML file Wizard used to create custom primitives that are debugged within CLIDE
- Simulator—described in the *CLIDE User's Manual*
- Source editor—described in the *CLIDE User's Manual*

<u>Source</u>: https://d3uzseaevmutz1.cloudfront.net/pubs/manual/cirrus_c_compiler_um15.pdf



9.2.2.1 Create Loops That Can Be Implemented as Hardware Loops

The CCC will translate to hardware loops only those loops for which it can calculate the number of iterations in compile-time. The loops that have a compile-time determinable number of iterations we will call "fixed-iterations loops". Of all possible fixed-iterations loops written in C, the CCC can only calculate the number of iterations for some cases. General guidelines for writing loops that the CCC can translate into hardware loops are:

- The loop boundaries must not change during the loop execution.
- The loop index must be updated only by addition or subtraction of a constant value.
- The loop index must not be updated in a conditional statement.

If you write a fixed-iterations loop which is not being translated to a hardware loop, turn on -emit-hints option, and see what the compiler reports for that loop. You should get precise reasons why the compiler cannot handle that particular loop as a hardware loop.

There are two groups of fixed-iterations loops. One group contains loops that have a constant number of iterations. Another group contains loops that have a number of iterations that varies between different evocations of the loop but is fixed once the loop starts execution (Example 9-8). The translating of the second group of loops to hardware loops is possible only when the -funsafe-loop-optimization option is on. That option lets the compiler assume that the number of iterations in such loop is never going to be 0. That assumption enables the compiler to safely translate those loops into hardware loops.

Here are several examples of loops that the CCC can successfully translate into hardware loops:

Example 9-4 Example 1 of loop that can be implemented as hardware loop by CCC

```
int foo ()
{
  int i, sum = 0;
  for (i = 0 ; i < 10 ; i++)
    sum += array1[i] + array2[i];
  return sum;
}
```

Source: https://d3uzseaevmutz1.cloudfront.net/pubs/manual/cirrus_c_compiler_um15.pdf

20. Defendant's actions complained of herein will continue unless Defendant is enjoined by this court.

21. Defendant's actions complained of herein are causing irreparable harm and monetary damage to Plaintiff and will continue to do so unless and until Defendant is enjoined and restrained by this Court.

22. Plaintiff is in compliance with 35 U.S.C. § 287.

PRAYER FOR RELIEF

WHEREFORE, Plaintiff asks the Court to:

(a) Enter judgment for Plaintiff on this Complaint on all causes of action asserted herein;

(b) Enter an Order enjoining Defendant, its agents, officers, servants, employees, attorneys, and all persons in active concert or participation with Defendant who receive notice of the order from further infringement of United States Patent No. 7,069,546 (or, in the alternative, awarding Plaintiff a running royalty from the time of judgment going forward);

(c) Award Plaintiff damages resulting from Defendant's infringement in accordance with 35 U.S.C. § 284;

(d) Award Plaintiff pre-judgment and post-judgment interest and costs; and

(e) Award Plaintiff such further relief to which the Court finds Plaintiff entitled under law or equity.

Dated: December 11, 2017

Respectfully submitted,

/s/ Jay Johnson

JAY JOHNSON
State Bar No. 24067322
D. BRADLEY KIZZIA
State Bar No. 11547550
KIZZIA JOHNSON, PLLC
1910 Pacific Ave., Suite 13000
Dallas, Texas 75201
(214) 451-0164
Fax: (214) 451-0165
jay@kjpllc.com
bkizzia@kjpllc.com

ATTORNEYS FOR PLAINTIFF

EXHIBIT A